Imperativ Programmering, tentamen

MARTIN MAGNUSSON
16 augusti 2017

- Inga hjälpmedel (böcker eller annat material) är tillåtna.
- Jourhavande lärare: Martin Magnusson. Martin kommer förbi cirka kl 9 för att svara på eventuella frågor.
- Tentan har tre delar, som svarar mot betygskriterierna för betyg 3, 4 och 5.
  - För att bli godkänd, med betyg 3, behövs 20 av 32 poäng från ”3”-frågorna (fråga 1–6) **eller** 32 poäng totalt (från alla frågor).
  - För betyg 4 behövs dessutom 10 av 16 från ”4”-frågorna (fråga 7–9) **eller** 22 poäng tillsammans från fråga 7–12.
  - För betyg 5 behövs dessutom 10 av 16 poäng från ”5”-frågorna (fråga 10–11).

Betyg 3

Fråga 1

8 poäng

Funktionen `sleep(unsigned int s)` pausar ett program i `s` sekunder. Hur många sekunder körs nedanstående kodsnuttar?

a)
```
for (int i = 0; i < 5; i++)
{
  sleep(1);
}
```

b)
```
for (int i = 0; i < 5; i++)
  sleep(1);
```

c)
```
for (int i = 0; i < 5; i = i+2)
{
  sleep(1);
}
```

d)
```
int i = 0;
while (i <= 5)
{
  sleep(1);
}
```

e)
```
int i = 0;
do
{
  sleep(1);
  i++;
} while (i < 5);
```

f)
```
int i = 1;
switch (i)
{
case 0:
  sleep(1);
  break;
case 1:
  sleep(1);
  break;
case 2:
  sleep(1);
  break;
default:
  sleep(1);
}
```

g)
```
int i = 1;
if (i > 0)
  sleep(1);
else
  sleep(2);
```

h)
```
int i = 0;
if (i > 0)
  sleep(1);
```

Fråga 2

4 poäng

Skriv en algoritm som, givet ett ord, skriver ut ordet ”med eko”. Exempel:

- eko → ekoko
- programmering → programmeringing
- Baloo → Balooloo
- packa → packaka

Du behöver alltså kolla om ordet slutar med konsonant eller inte, och beroende på det skriva ut några av de sista tecknen igen. (Om det sista tecknet är en konsonant behöver du leta fram till den sista vokalen, och vice versa.) Du kan anta att det finns färdiga funktioner som kan avgöra om ett tecken är en vokal eller konsonant.

Du kan antingen skriva din algoritm med C-syntax eller på svenska (t ex som en punkt-lista). Om du väljer att skriva på svenska är det viktigt att funktionsbeskrivningen ändå är lika detaljerad som den skulle behöva vara i ett C-program. Däremot behöver inte syntaxen vara enligt C. (Det är antagligen lättare att skriva lösningen på svenska, eftersom stränghantering i C kan vara knökigt.)

För den här uppgiften ges inga avdrag för syntaxfel, så länge det är klart vad som är meningen ska hända. Det är inte en övning i C-programmering, men däremot i algoritmiskt tänkande.

Fråga 3

4 poäng

Vilket värde har `x` i slutet av följande kodsnuttar?

a)
```
int a = 1;
int x = (a != 0);
```

b)
```
int a = 1;
float x = a;
```

c)
```
int x = 0;
int y = 1;
{
  int x = 10;
  int y = 20;
  x += y;
}
x += y;
```

d)
```
unsigned char a = 100;
unsigned char b = 260;
int x = (a < b);
```

Fråga 4

8 poäng

Lista alla rader det är fel på i följande program, och vad det är för fel. Det är fel på åtta rader totalt. (Det finns både syntaxfel, som inte går igenom kompilatorn, och logiska fel, som ger fel resultat.)

Skriv inte ditt svar direkt på tentan, utan på ett lösningsblad, precis som med de andra frågorna.

Eventuellt finns det saker i programmet som är riskabla eller osnygga, men i den här uppgiften ska du bara markera det som faktiskt blir *fel*.


```c
#include stdio.h
#include <string.h>

/**
 * Check if a string is a palindrome or not. That is, check if it
 * reads the same from the left and the right.
 *
 * @param string Input string to test.
 *
 * @return 1 if the string is a palindrome, and 0 otherwise.
 */
int palindrome char *string
{
  int n = strlen( string );
  for (int i = 0; i < n/2; i == i+1)
  {
    if (string[i] != string[n-i-1]) // jämför första/sista tecknet o.s.v.
      0;
  }
  1;
}

int main( int argc, char *argv[] )
{
  if (argc < 1)
  {
    if (palindrome( argv[1] ))
    then
      printf( "%s är ett palindrom\n", argv[1] );
    else
      printf( "%s är inte ett palindrom\n", argv[1] );
    endif
  }
  return 0;
}
```

Fråga 5

4 poäng

Den här uppgiften handlar om pekare och fält.

a) Hur många element innehåller fältet nedan?

```c
float a[10];
```

b) Vilken typ ska funktionen `f` ta som inargument i det här exemplet?

```c
int *b;
*b = 10;
f(&b);
```

c) Hur många floats innehåller fältet nedan?

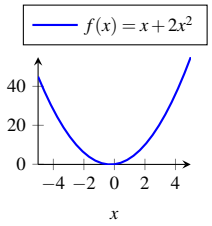
```c
float c[10][10];
```

d) Vilken typ ska funktionen g ta som inargument i det här exemplet?

```c
int *d;
*d = 10;
g(*d);
```

Fråga 6

4 poäng



Anta att du får en programbibliotek som innehåller följande funktionsdeklaration.

```
/**
 * Compute the value of a polynomial for a particular value of its free
 * variable. E.g., if f is x^2 and x is 0.5, the function returns
 * 0.25.
 *
 * @param f The polynomial (for example: x^2, 3*x^2+4*x, 10).
 * @param x The value of f's free variable. If the polynomial does not
 *    have a free variable, the value of x is ignored.
 * @return The value of f(x).
 */
float eval_function( FunctionPtr f, float x );
```

Du har inte tillgång till funktionsdefinitionen, som ligger i en kompilerad objektfil.

Ange ett antal testfall (minst 4) för att testa att funktionen verkligen gör det som sägs i dokumentationen. Se till att dina testfall täcker olika tänkbara felkällor.

Ange både testfall och förväntat resultat. Du behöver inte skriva ditt svar i C (eftersom det inte är självklart hur `FunctionPtr` ser ut), utan lista i stället ett antal funktioner (polynom) med x-värde, och förväntat resultat.

- $f = \ldots$, $x = \ldots$, förväntat resultat $f(x) = \ldots$
- $\ldots$

Betyg 4

Fråga 7

Strängar i C brukar representeras med ett char-fält – som har fix storlek – med tecknet `\0` som avslutning.

Tänk dig att ditt program behöver mer flexibel lagring av strängar, och du har därför bestämt dig för att implementera en abstrakt datatyp där strängar lagras som länkade listor i stället.

2 poäng

a) Skriv den **`struct`** som är den konkreta implementationen för ett element av din länkade lista.

4 poäng

b) Skriv en funktion `add_after` som lägger in ett nytt listelement efter ett annat. Argument till den här funktionen ska vara en pekare till elementet där något ska läggas efteråt, och tecknet som ska ligga i det elementet.

4 poäng

c) Skriv en funktion som använder din datastruktur och som översätter en sträng till ”rövarspråk”. Det betyder att varje konsonant ’x’ byts ut mot ’xox’. T ex, ”hej” → ”hohejoj”, och ”Örebro” → ”Örorebobroro”. För att göra detta behöver du alltså lägga till nya element i mitten av din lista med tecken.

Du kan anta att det finns en funktion **`int`** `is_consonant(`**`char`** `c)` som returnerar 1 om dess parameter är en konsonant, och 0 annars.

Fråga 8

Din abstrakta datatyp från föregående uppgift 7a har nu en hjälpfunktion `add_after` för att lägga in ett element mitt i listan. Dessutom kan man tänka sig att du i samma fil implementerar även andra tillhörande funktioner som sköter om hur listan skapas och hur den används (t ex att initiera en tom sträng, och att ta bort tecken). Däremot finns det inget som hindrar en användare som läser källkoden för din implementation att strunta i hjälpfunktionerna som du har angett (t ex `add_after`) och i stället direkt modifiera **`struct`**-en från uppgift 7a.

2 poäng

a) Beskriv varför det kan leda till problem.

2 poäng

b) Beskriv hur du kan göra för att undvika det, men samtidigt erbjuda samma gränssnitt som definierades i uppgift 7.

Fråga 9

2 poäng

Se nedanstående två programskelett som båda skapar en sekvens med 100 heltal, och sedan plockar ut det nittionde talet. (För program 2 får du förutsätta att det finns en `create_list` som skapar en lagom stor lista och returnerare en pekare till det första elementet.)

a) Vilket av de två implementationerna av `get_n` bör vara snabbast? Motivera ditt svar.

b) Vilken implementation är mest minneseffektiv om det under programmets körning behövs talsekvenser av olika längd, så att det ibland räcker med ett tal, och ibland krävs 100 000? Motivera ditt svar.

Program 1:

```
int numbers[100];

int get_n
( int *numbers, int n )
{
  ...
}

// Get the 90th number:
int n90 = get_n( numbers, 90 );
```

Program 2:

```
int_list *numbers;

numbers = create_list( 100 );

int get_n
( int_list *numbers, int n )
{
  ...
}

// Get the 90th number:
int n90 = get_n( numbers, 90 );
```

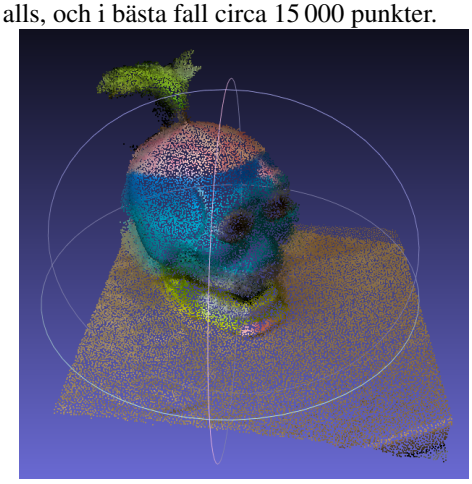
Betyg 5

Fråga 10

Det har nyligen börjat komma mobiltelefoner som har inbyggda 3D-kameror, och alltså kan skapa en tredimensionell modell av objekt och rum. Till skillnad från vanliga 2D-kameror, där datat består av ett 2D-rutnät med pixlar, med ett fast antal pixlar i varje bild, så hanteras data från dessa kameror ofta i form av *punktmoln*, det vill säga en mängd 3D-punkter som inte är ordnade i ett fixt rutnät.

Beroende på hur omgivningen ser ut kan det komma olika antal punkter i en 3D-bild. I ett mörkt rum kanske det inte kommer några alls, och i bästa fall circa 15 000 punkter.

En 2D-bild.



Ett punktmoln.

3 poäng

a) Implementera i C en datastruktur eller datastrukturer för att representera ett punktmoln. Anta att en punkt bara behöver information om positionen (x, y, z) och inga andra data, som färg, etc. Det räcker med att beskriva en implementation av själva datastrukturen, och inte några hjälpfunktioner.

1 poäng

b) Vilka ändringar behöver du göra i din implementation för att hantera färgade punktmoln, där varje punkt förutom position också vet vad det är för färg där?

4 poäng

c) Diskutera också din implementation för *punkter* jämfört med minst en annan tänkbar lösning. Vad har de för för- och nackdelar?

4 poäng

d) Diskutera också din implementation för *punktmoln* jämfört med minst en annan tänkbar lösning. Vad har de för för- och nackdelar?

För de här diskussionsfrågorna gäller att åtminstone två lösningar (varav den ena kan vara den lösning du har använt ovan) ska jämföras, och att det framgår att du förstår hur de fungerar och deras för- och nackdelar.

Fråga 11

När man gör plats för data i ett program så kan man antingen göra det statiskt (det som ligger på en funktions aktiveringspost, eller *stack frame*) eller dynamiskt.

2 poäng

a) Beskriv en skillnad mellan att allokera minne statiskt och dynamiskt i en funktion i C. Gör det någon skillnad för kod som anropar den funktionen? Motivera ditt svar.

2 poäng

b) Gör programmet nedan det som sägs i kommentarerna? Motivera ditt svar.

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  int main( int argc, char *argv[] )
4  {
5    char *s = malloc( 10 * sizeof(char) );
6    s[0] = 'A';
7    printf("%c\n", s[0]); // Skriver ut 'A'
8    free( s );
9    char A = s[0];
10   printf("%c\n", A);    // Skriver ut 'A' igen
11
12   return 0;
13 }
```